Introduction à R

Christophe Lalanne & Bruno Falissard

Table des matières

1 Introduction

Ce document constitue une présentation succincte des bases du langage R pour l'analyse statistique interactive de données. En particulier, on s'intéressera à la représentation et la manipulation des données numériques et qualitatives, à l'importation de source de données externes et à la sauvegarde d'une session de travail. Un glossaire

des principales commandes est également fourni en Annexe, ainsi qu'une liste des commandes utiles pour la modélisation.

Le logiciel R est disponible pour Windows, Mac et Linux, et son installation ne présente en règle générale aucune difficulté. Pour plus d'informations concernant l'installation du logiciel, il peut être utile de consulter la FAQ, en particulier la section How can R be installed ?.

L'interface de R est assez rudimentaire, et diffère des logiciels tels que Stata, SPSS ou Statistica qui offrent une vue des données comme sous un tableur (par exemple, Microsoft Excel) et des menus déroulants. Sous R, l'utilisateur écrit explicitement des commandes permettant de travailler sur les données. Notons que le logiciel RStudio fournit une interface plus conviviale que l'interface de base de R.

En termes de guide pour démarrer, le site CRAN héberge l'aide en ligne officielle, dont An Introduction to R, qui décrit les principales commandes de R. La section "Contributed" du site CRAN propose également des documents en français [1].

2 Interagir avec R

> **Démarrer avec R.** Quelque soit le système d'exploitation utilisé (Windows, Mac, Linux), R fonctionne comme tout autre logiciel : il suffit généralement de double-cliquer sur l'icône de l'application pour démarrer R. On dispose ensuite d'une console interactive dans laquelle on peut commencer à saisir des commandes après l'invite R >. Les résultats seront affichés aussitôt dans la console.

2.1 Interactivité et reproductibilité

R est avant tout un langage et un interpréteur de commandes. L'approche est interactive dans la mesure où il est possible de taper directement des commandes à l'invite R et de visualiser le résultat dans la même interface. On parlera de "console" pour désigner la fenêtre interactive dans laquelle on saisit des commandes R et où l'on visualise les résultats renvoyés par R. Les graphiques sont générés dans une fenêtre graphique externe.

Il est également possible d'enregistrer une série de commandes dans un fichier script R, ayant pour extension `.R` ou `.r`, et de faire exécuter l'intégralité des commandes de ce script par R à l'aide de la commande `source()`. Quelle que soit la plateforme, R fournit un éditeur minimal qui offre la possibilité d'envoyer interactivement des commandes dans la console, ou l'intégralité des commandes d'un script, à l'image de `source()`.

2.2 Obtenir de l'aide

Le système d'aide en ligne fourni avec R est accessible via la commande `help()`. Lorsque l'on connaît le nom de la commande R, par exemple, `cmd()`, on peut taper `help(cmd)` ou `?cmd` (sauf dans le cas de certains opérateurs). Sinon, on peut rechercher à partir de mots-clés en tapant `help.search(cmd)`. Une alternative pour la recherche par motif consiste à utiliser `apropos(cmd)`. Pour connaître toutes les commandes fournies par un package (e.g., `pkg`), il suffit de taper `help(package=pkg)`.

1. http://cran.r-project.org/other-docs.html

3 Représentation des données sous R

3.1 Variables numériques et catégorielles

> **Les variables sous R.** Formellement, une telle liste de nombres est stockée dans ce que R appelle un vecteur. Par souci de simplicité, nous parlerons de variable et d'éléments d'une variable. Sur le plan statistique, les éléments seraient plutôt considérés comme des observations ou des données collectées sur chaque unité statistique.

Supposons que l'on ait demandé à 10 personnes choisies au hasard dans la rue leur âge. Voici la série de mesures recueillies, arrondies à l'entier le plus proche, ainsi que le sexe de la personne.

age	18	27	34	18	24		30	28	19	19
sexe	F	F	M	F	M	M	M	F	M	F

Dans l'exemple suivant, on crée une variable appelée `age` (il n'est pas recommandé d'utiliser des accents ou signes diacritiques) à laquelle on associe la liste des nombres présentés dans le tableau précédent.

```
age <- c(18, 27, 34, 18, 24, NA, 30, 28, 19, 19)
```

Pour assigner un nom de variable à une série de valeurs, on utilise le symbole `<-`. Le signe = est valide également, mais il n'est pas recommandé de l'utiliser dans ce contexte. Les valeurs sont listées à l'intérieur d'une commande `c()`, entre parenthèses. Toutes les commandes R utilisent le même principe : les données ou les options se trouvent mentionnées entre parenthèses. La sixième personne interrogée ayant refusé de répondre, on considère qu'il s'agit d'une valeur manquante que nous avons représentée par un point (`.`), représentée sous R par le symbole `NA`. Pour afficher le contenu de la variable `age`, il suffit de taper son nom :

```
age
```

```
##  [1] 18 27 34 18 24 NA 30 28 19 19
```

En ce qui concerne le stockage de la deuxième série de mesures (variable `sexe`), on notera qu'il ne s'agit pas d'une variable numérique, mais d'une série de lettres {F, M}. Il serait tout à fait possible de considérer un codage numérique pour cette variable, en décidant de représenter les femmes par des 0 et les hommes par des 1. On peut toutefois créer une variable constituée de caractères de la manière suivante :

```
sexe <- c("F", "F", "M", "F", "M", "M", "M", "F", "M", "F")
sexe
```

```
##  [1] "F" "F" "M" "F" "M" "M" "M" "F" "M" "F"
```

Sous R, les caractères ou chaînes de caractères sont entourés de guillemets anglo-saxons, simple ou double "quote".

> **Nommage de variables.** On retiendra également que les noms de variable sont sensible à la casse : la variable `sexe` est différente d'une variable qui serait appelée `Sexe`.

Par souci de simplicité, on ne fera pas de distinction entre les variables au sens statistique du terme, et les variables sous R. On considérera donc qu'une variable R possède un nom et contient une série de valeurs de type numérique (entier ou nombre réel) ou caractère, ce que l'on peut vérifier généralement à l'aide de `mode()`. Le nombre total d'éléments contenus dans une variable, incluant les éventuelles données manquantes `NA`, est obtenu avec la commande `length()`.

```
length(age)
```

```
## [1] 10
```

```
length(sexe)
```

```
## [1] 10
```

```
mode(age)
```

```
## [1] "numeric"
```

```
mode(sexe)
```

```
## [1] "character"
```

Supposons qu'une autre variable ait été collectée, à savoir la réponse concernant le degré d'accord des répondants vis-à-vis d'une certaine assertion (par exemple, "quelle est votre opinion concernant les théories selon lesquelles le climat se réchauffe et entraînera à terme de grosses difficultés pour vivre sur Terre"). Les 5 modalités de réponse proposées aux participants suivent le principe d'une échelle de Likert [2]. Les données ont été recueillies au format numérique, de 1="Pas du tout d'accord" à 5="Tout à fait d'accord". Appelons cette variable `opin`.

```
opin <- c(1, 3, 2, 1, 4, 1, 5, 3, 2, 2)
```

La commande `factor()` permet de faire connaître à R la nature qualitative de cette variable, et d'associer à chaque modalité ou niveau du facteur (`levels`) des étiquettes textuelles plus informatives. Par défaut, la commande `factor()` ne fait qu'ajouter des niveaux à une variable, les niveaux retenus correspondant aux valeurs uniques, triées par ordre lexicographique, présentes dans la variable.

```
factor(opin)
```

```
##  [1] 1 3 2 1 4 1 5 3 2 2
## Levels: 1 2 3 4 5
```

Il est possible d'associer des étiquettes à chacun de ces niveaux, en respectant l'ordre de présentation des niveaux à l'aide de l'option `labels=`. La commande `nlevels()` renverra le nombre de niveaux d'une variable de type `factor`.

2. http://fr.wikipedia.org/wiki/Échelle_de_Likert

```
opin <- factor(opin, labels=c("Pas du tout d'accord", "Moyennement d'accord",
                              "Sans Opinion", "Assez d'accord", "Tout a fait d'accord"))
opin
```

```
##  [1] Pas du tout d'accord Sans Opinion         Moyennement d'accord
##  [4] Pas du tout d'accord Assez d'accord       Pas du tout d'accord
##  [7] Tout a fait d'accord Sans Opinion         Moyennement d'accord
## [10] Moyennement d'accord
## 5 Levels: Pas du tout d'accord Moyennement d'accord ... Tout a fait d'accord
```

```
nlevels(opin)
```

```
## [1] 5
```

La commande `levels()` permet de lister les niveaux d'une variable qualitative, ou d'en modifier les valeurs. L'exemple ci-dessous montre comment il est possible d'agréger les deux dernières modalités de la variable.

```
levels(opin)
```

```
## [1] "Pas du tout d'accord" "Moyennement d'accord" "Sans Opinion"
## [4] "Assez d'accord"       "Tout a fait d'accord"
```

```
levels(opin)[4:5] <- "Assez ou tout a fait d'accord"
opin
```

```
##  [1] Pas du tout d'accord          Sans Opinion
##  [3] Moyennement d'accord          Pas du tout d'accord
##  [5] Assez ou tout a fait d'accord Pas du tout d'accord
##  [7] Assez ou tout a fait d'accord Sans Opinion
##  [9] Moyennement d'accord          Moyennement d'accord
## 4 Levels: Pas du tout d'accord Moyennement d'accord ... Assez ou tout a fait d'accord
```

Il est possible d'indiquer à R que les niveaux de la variable qualitative sont ordonnées (on parle parfois de variable ordinale, contrairement aux variables nominales dont les modalités ne sont pas ordonnées, comme la couleur des yeux par exemple).

```
factor(opin, ordered=TRUE)
```

```
##  [1] Pas du tout d'accord          Sans Opinion
##  [3] Moyennement d'accord          Pas du tout d'accord
##  [5] Assez ou tout a fait d'accord Pas du tout d'accord
##  [7] Assez ou tout a fait d'accord Sans Opinion
##  [9] Moyennement d'accord          Moyennement d'accord
## 4 Levels: Pas du tout d'accord < ... < Assez ou tout a fait d'accord
```

La variable `sexe` manipulée précédemment pourrait tout aussi bien être convertie en facteur :

```
sexe <- factor(sexe)
levels(sexe)
```

```
## [1] "F" "M"
```

On notera que le premier niveau est `F`. Si l'on souhaite que celui-ci soit `M`, il suffit d'utiliser la commande `relevel()` et d'indiquer la catégorie de référence en option.

> **Les facteurs sous R.** Les niveaux des facteurs sont déterminés à partir des valeurs uniques identifiées dans une variable. L'ordre des niveaux d'un facteur suit l'ordre lexicographique : nombres, lettres en minuscules/majuscules. Par exemple, `factor(c("1", "oui", "Oui", "Non", "non", "3"))`. R supprime automatiquement les valeurs manquantes de la liste des niveaux, sauf si l'on indique l'option `exclude=NULL` ; dans ce cas, les valeurs manquantes figureront comme un niveau à part.

```
sexe <- relevel(sexe, ref="M")
levels(sexe)
```

```
## [1] "M" "F"
```

3.2 Indexation d'observations

Il existe deux moyens d'accéder aux éléments d'une variable : renseigner le ou les numéros d'observation ou utiliser un filtre ou une condition à vérifier. On appellera cette dernière approche l'indexation sur critères.

Pour accéder individuellement aux éléments d'une variable, on écrit le nom de la variable suivi du numéro d'élément entre crochets. La première observation de la variable `age` vaut ainsi :

```
age[1]
```

```
## [1] 18
```

Il est possible de désigner plusieurs éléments, en les insérant dans une liste via `c()`.

```
sexe[c(1, 2, 3)]
```

```
## [1] F F M
## Levels: M F
```

Lorsque les éléments que l'on souhaite afficher sont successifs (ici, les observations 1 à 3), L'expression ci-dessus peut se simplifier en utilisant la notation ci-dessous, dans laquelle on indique l'élément de départ et l'élément d'arrivée. L'expression ci-dessus est donc strictement équivalente à `sexe[1:3]`.

> **Motifs réguliers.** R dispose de deux commandes, `rep()` et `seq()`, qui permettent de générer des séquences de nombres suivant un motif particulier. Par exemple, `seq(1, 10, by = 2)` renvoie la liste des 5 premiers nombres impairs, alors que `rep(c(1, 3), 2)` répète la liste `(1,3)` deux fois.

On peut vérifier la présence de données manquantes grâce à `is.na()`, ce qui nous permet également d'introduire un autre type de données R : les variables booléennes, c'est-à-dire des variables ne prenant que deux valeurs, vrai (`TRUE`) ou faux (`FALSE`), et que l'on retrouvera dans les tests logiques.

```
is.na(age)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
```

Le résultat produit par R est de même longueur que la variable `age` puisque R teste si chaque élément de cette variable est égal à `NA`. L'expression `age == NA`, où le symbole == désigne le test d'égalité logique (à ne pas confondre avec =) produirait le même résultat. Puisque la commande `which()` renvoie le numéro du ou des éléments remplissant une certaine condition logique, on peut combiner `is.na()` et `which()` pour obtenir le numéro de l'observation manquante dans la variable `age`.

```
which(is.na(age))
```

```
## [1] 6
```

Les observations disponibles pour ces deux variables ont été recueillies sur la même unité statistique. Il est donc naturel de se demander quel est le sexe de la personne n'ayant pas renseigné son âge. Plutôt que `sexe[6]`, on pourra préférer utiliser l'expression suivante :

```
sexe[which(is.na(age))]
```

```
## [1] M
## Levels: M F
```

> **Tests logiques sous R.** Les opérateurs logiques sous R sont les suivants : == (égalité), != (non égalité), ! (négation), & (et), | (ou), > (supérieur à), >= (supérieur ou égal à), < (inférieur à), <= (inférieur ou égal à).

Un autre exemple d'indexation critériée consisterait à lister le sexe des individus dont l'âge est supérieur à 25 ans, par exemple. Le test logique à vérifier est alors : `age > 25`, ou `age >= 25` si l'on souhaite inclure la valeur 25 et toutes les observations vérifiant cette condition seront inclues dans le résultat renvoyé par R. Voici un exemple d'application :

```
sexe[which(age > 25)]
```

```
## [1] F M M F
## Levels: M F
```

3.3 Commandes R et opérations sur des variables

Un aspect important de R est que les commandes opèrent généralement sur l'ensemble des éléments d'une variable. Il n'est pas nécessaire de construire des boucles pour itérer une opération sur chacun des éléments.

La plupart des opérations arithmétiques de base sont disponibles sous R : addition, soustraction, division, etc. Par exemple `1 + 2` renverra 3. Il faut toutefois faire attention à la représentation des nombres réels en machine… R offre par ailleurs des commandes permettant de travailler avec l'ensemble des éléments d'une variable, comme par exemple `sum()` qui renvoie la somme des valeurs stockées dans une variable numérique :

```
sum(age)
```

```
## [1] NA
```

On voit que le résultat renvoyé ne correspond pas vraiment au résultat escompté, et que R se contente de renvoyer la valeur `NA`. Comme il y a une valeur manquante, R ne sait pas comment on doit la traiter et rappelle à l'utilisateur que la série de valeurs dont on essaye de calculer la somme n'est pas complètement observée. On peut alors effectuer le calcul en ignorant la 6eme observation.

```
sum(age[-6])
```

```
## [1] 217
```

Mais en fait la commande `sum()` dispose d'une option permettant d'effectuer le calcul en ignorant les valeurs manquantes.

```
sum(age, na.rm=TRUE)
```

```
## [1] 217
```

Il est bien entendu possible d'appliquer des opérations de manière séquentielle, par exemple calculer le carré des âges des individus, puis la somme de ces carrés, mais ce type d'opération peut être réalisé en une seule fois comme le montre l'exemple suivant :

```
sum(age^2, na.rm=TRUE)
```

```
## [1] 5515
```

Dans l'exemple suivant, on montre comment, après avoir remplacé la donnée manquante pour l'âge par la moyenne des âges de l'échantillon, il est possible de réaliser des opérations arithmétiques simples, qui opèrent toujours sur l'ensemble des éléments manipulés.

```
age[is.na(age)] <- mean(age, na.rm=TRUE)
n <- length(age)      ## nombre d'observations
age - mean(age)       ## écarts à la moyenne
```

```
##  [1] -6.1111  2.8889  9.8889 -6.1111 -0.1111  0.0000  5.8889  3.8889
##  [9] -5.1111 -5.1111
```

```
(age - mean(age))^2   ## carrés des écarts à la moyenne
```

```
##  [1] 37.34568  8.34568 97.79012 37.34568  0.01235  0.00000 34.67901
##  [8] 15.12346 26.12346 26.12346
```

```
sum((age - mean(age))^2)/(n-1)
```

```
## [1] 31.43
```

```
sqrt(sum((age - mean(age))^2)/(n-1))
```

```
## [1] 5.606
```

Évidemment, inutile de calculer un écart-type manuellement puisque R dispose déjà d'une commande `sd()` pour réaliser ce calcul :

```
sd(age)
```

```
## [1] 5.606
```

Des commandes similaires et très utiles lorsqu'il s'agit de résumer la distribution d'une variable numérique à l'aide des principaux indicateurs de tendance centrale et de dispersion sont : `range()` (`min()` et `max()`), `mean()`, `median()`, `var()`, `IQR()`. Plus généralement, la commande `summary()` fournit de manière compacte la plupart de ces indicateurs concernant la distribution d'une variable numérique, ou un tableau d'effectifs dans le cas d'une variable qualitative. La commande `summary()` fonctionne avec une ou plusieurs variables (voir section suivante).

Notons que tous les résultats renvoyés par R peuvent être stockés dans des variables également, ce qui permet de sauvegarder des résultats intermédiaires, ou de les réutiliser plus tard.

```
res <- sum(age, na.rm=TRUE)
res
```

```
## [1] 241.1
```

On peut visualiser l'ensemble des variables présentes dans l'espace de travail ("workspace") grâce à `ls()`.

```
ls()
```

```
## [1] "age"      "d"        "metadata" "n"        "opin"     "res"
## [7] "sexe"
```

Suppression de variables. On retiendra que toute altération permanente ou suppression de variables est définitive. Les données d'origine sont définitivement enlevées de l'espace de travail.

La commande `rm()` permet de supprimer une ou plusieurs variables de l'espace de travail.

```
rm(n, res)
ls()
```

```
## [1] "age"      "d"        "metadata" "opin"     "sexe"
```

3.4 Tableaux de données hétérogènes

Plutôt que de travailler avec des variables isolées comme précédemment, lorsque les données ont été collectées sur les mêmes unités statistiques il est plus intéressant de constituer une véritable structure de données, appelée data frame sous R. Il s'agit d'un tableau rectangulaire dans lequel les variables sont arrangées en colonnes et les observations en lignes, et les variables peuvent être de différents types, numériques ou catégorielle, contrairement à des structures de données plus simples définies par la commande `matrix()`, par exemple [3].

La commande `data.frame()` s'utilise de manière assez simple : on lui fournit la liste des variables à inclure dans le tableau de données. Éventuellement, les variables peuvent être renommées directement lors de la construction du data frame, comme illustré dans l'exemple suivant.

```
d <- data.frame(age, sex=sexe)
d
```

```
##      age sex
## 1  18.00   F
## 2  27.00   F
## 3  34.00   M
## 4  18.00   F
## 5  24.00   M
## 6  24.11   M
## 7  30.00   M
## 8  28.00   F
## 9  19.00   M
## 10 19.00   F
```

Un data frame comprend des dimensions (`dim()` fournit le nombre de lignes et le nombre de colonnes), des noms de variables (`names()`) et des numéros d'observations, `rownames()` (qui servent d'identifiants uniques), et une description du type de variables présentes dans le data frame est obtenue à l'aide de la commande `str()`.

```
dim(d)
```

```
## [1] 10  2
```

3. Les objets R de type `matrix` ne peuvent contenir que des données du même type, par exemple trois variables numériques arrangées en colonnes.

```r
names(d)
```

```
## [1] "age" "sex"
```

```r
str(d)
```

```
## 'data.frame':    10 obs. of  2 variables:
##  $ age: num  18 27 34 18 24 ...
##  $ sex: Factor w/ 2 levels "M","F": 2 2 1 2 1 1 1 2 1 2
```

Pour accéder à une variable spécifique dans un data frame, on utilisera le nom du data frame, suivi du symbole `$` et du nom de la variable. Ainsi l'expression `d$age` peut se lire comme "la variable age dans le data frame d". Le principe d'indexation des variables vu à la section précédente reste applicable dans le cas des data frame, naturellement. Cependant, comme il s'agit d'un tableau, l'indexation des observations se fait à partir des lignes du tableau : on notera `[i,j]` la *i*eme observation pour la jeme variable. Ainsi, `d[1,1]` désignera l'âge du premier individu, `d[c(1,2,3),1]` l'âge des 3 premiers individus, et `d[1:2,1:2]` renverra toutes les données recueillies sur les deux premiers individus. Dans ce dernier cas, il n'est pas nécessaire de préciser les numéros de colonne si elles sont toutes inclues dans la sélection ; on pourra donc écrire `d[1:2,]` au lieu de `d[1:2,1:2]`.

```r
d$age
```

```
##  [1] 18.00 27.00 34.00 18.00 24.00 24.11 30.00 28.00 19.00 19.00
```

```r
d$age[1:2]
```

```
## [1] 18 27
```

```r
d[1:2, 1]
```

```
## [1] 18 27
```

Puisque les colonnes d'un data frame sont nommées, il est tout à fait possible de remplacer l'expression `d[1:2,1]` par `d[1:2,"age"]`, ce qui facilite la relecture, ou permet de bien travailler sur la variable `age` en cas de changement de position des variables dans le data frame.

Il est également possible d'ajouter ou de supprimer des variables dans un data frame :

```r
d$var1 <- 1:10
d[1:3,]
```

```
##   age sex var1
## 1  18   F    1
## 2  27   F    2
## 3  34   M    3
```

```
d[,3] <- NULL
d[1:3,]
```

```
##   age sex
## 1  18   F
## 2  27   F
## 3  34   M
```

```
d$opinion <- opin
str(d)
```

```
## 'data.frame':    10 obs. of  3 variables:
##  $ age    : num  18 27 34 18 24 ...
##  $ sex    : Factor w/ 2 levels "M","F": 2 2 1 2 1 1 1 2 1 2
##  $ opinion: Factor w/ 4 levels "Pas du tout d'accord",..: 1 3 2 1 4 1 4 3 2 2
```

Comme on l'a dit plus haut, il est également possible d'utiliser `summary()` pour obtenir un résumé détaillé de la distribution de chaque variable d'un data frame.

```
summary(d)
```

```
##       age       sex                              opinion
##  Min.   :18.0   M:5   Pas du tout d'accord          :3
##  1st Qu.:19.0   F:5   Moyennement d'accord           :3
##  Median :24.1         Sans Opinion                   :2
##  Mean   :24.1         Assez ou tout a fait d'accord:2
##  3rd Qu.:27.8
##  Max.   :34.0
```

4 Importer des données

4.1 Données texte simple

La commande `read.table()` permet d'importer sous R des données stockées dans un format proche de celui du data frame, c'est-à-dire où les variables sont arrangées en colonnes et les observations en ligne. Voici un exemple d'aperçu des données précédentes lorsqu'elles sont sauvegardées dans un tel format (seules les trois premières lignes du fichier sont affichées). Chaque valeur est séparée par un espace : il s'agit du séparateur de champ. La première ligne du fichier `donnees.txt` contient le nom des variables. Toutes les chaînes de caractères sont entourées de quotes anglo-saxonnes.

```
"age" "sexe" "opinion"
18 "F" "Pas du tout d'accord"
27 "F" "Sans Opinion"
```

Pour importer ce fichier, on utilisera une instruction du type :

```
d <- read.table("donnees.txt", header = TRUE, sep = "", dec = ".")
```

Certaines des options par défaut ont été renseignées explicitement, comme par exemple le délimiteur décimal (`dec=`). Il est surtout important d'indiquer à R si le fichier contient une ligne d'en-tête ou pas (`header=`).

Dans les cas plus simples où il n'y a qu'une série de valeurs à importer sous R, la commande `scan()` fonctionne sur le même principe.

4.2 Données au format CSV

Les données exportées depuis un tableur de type Excel au format CSV peuvent être importées sous R à l'aide de la commande `read.csv()`, si le séparateur de champ est une virgule, ou `read.csv2()`, si le séparateur de champ est un point-virgule. Voici un exemple d'aperçu des données précédentes lorsqu'elles sont sauvegardées dans un tel format (seules les trois premières lignes du fichier sont affichées). Ici, le séparateur de champ est une virgule (dans ce cas, le séparateur décimal est obligatoirement un point).

```
"age","sexe","opinion"
18,"F","Pas du tout d'accord"
27,"F","Sans Opinion"
```

Pour importer ce fichier, on utilisera une instruction du type :

```
d <- read.csv("donnees.csv")
```

Que ce soit avec `read.table()` ou `read.csv()`, les données manquantes sont généralement considérées des valeurs absentes dans le fichier. Tout autre symbole signalant des valeurs manquantes doit être explicitement renseigné dans l'option `na.strings=`.

> **XLS ou CSV.** Il est également possible de lire directement des fichiers Excel, voire une sous-partie d'une feuille de calcul (par exemple, une zone de plage `A2:C5`, soit 12 cellules au total), à l'aide de packages spécialisés. Cela dit, comme il est tout aussi simple d'exporter les données au format CSV depuis Excel, et que ce type de format de données pose moins de problème de compatibilité de version entre les logiciels et les systèmes d'exploitation, on préférera généralement le format CSV.

4.3 Données enregistrées à partir d'autres logiciels statistiques

Le package **foreign** contient des commandes permettant d'importer des sources de données enregistrées à partir d'autres logiciels statistiques. En particulier, les données au format SPSS, Stata et SAS peuvent chargées sous R à l'aide des commandes `read.spss()` (dans ce cas, il ne faut surtout pas oublier de rajouter l'option `to.data.frame = TRUE`), `read.dta()` et `read.xport()`.

5 Installer des packages additionnels

> **Installation de packages.** Il est également possible d'installer des packages en utilisant les menus déroulant de l'interface R. Il suffit généralement d'indiquer le nom du ou des packages à installer, et de choisir un site pour le téléchargement des packages (il est conseillé de choisir un serveur proche de l'endroit où l'on se trouve pour améliorer la rapidité de téléchargement).

R offre l'essentiel des commandes permettant de décrire, modéliser et représenter graphiquement des jeux de données multivariables. Il existe cependant tout un écosystème de commandes additionnelles, regroupées par thème dans ce que l'on appelle des packages. La liste complète de ces packages est disponible sur le site CRAN, mais il existe également des lots de packages regroupés par domaine (graphiques, plans d'expérience, sondages, données géospatiales, etc.) que l'on trouvera sur la page Task View.

Pour installer un package, la commande à utiliser est `install.packages()`, et souvent il sera nécessaire d'ajouter l'option `dependencies = "Depends"` si le package à installer dépend de commandes fournies dans d'autres packages externes. Dans ce cours, par exemple, les packages **prettyR** et **gplots** seront utilisés et pourront être installés de la manière suivante.

```
install.packages(c("prettyR", "gplots"), dep = "Depends")
```

Pour connaître l'ensemble des commandes disponibles dans un package, il suffit de taper l'instruction suivante :

```
help(package=MASS)
```

6 Les fonctionnalités graphiques

R offre trois principaux systèmes graphiques : le système graphique de base, utilisé dans le cadre de ce cours, lattice et ggplot2.

Le chapitre 4 du manuel R pour les débutants de E. Paradis fournit un aperçu relativement complet des différentes commandes graphiques et des outils de personnalisation et de gestion des fenêtres graphiques.

7 Les outils statistiques

Ci-dessous figure une liste des principales commandes R pour les mesures et tests d'association entre 2 ou plusieurs variables numériques et/ou qualitatives. Pour chaque commande, seules les options obligatoires ou essentielles sont présentées. Les autres sont à rechercher dans l'aide en ligne. Dans tous les cas, les variables `x` et `y` sont considérées comme des variables numériques, `z` est une variable qualitative ou facteur (de même que `z1` et `z2`), `s` une variable censurée (à droite) et construite à l'aide de la commande `Surv()`, et `cmd` désigne n'importe quelle commande R permettant d'opérer sur une variable numérique, par exemple `mean`. Enfin, `mod` et `tab` désignent respectivement le résultat d'un modèle linéaire ou d'un tableau de contingence stocké dans une variable auxiliaire.

7.1 Comparaison de moyennes

Commande	Description
`tapply(x, z, cmd, ...)`	Application d'une commande `cmd` à une variable numérique `x` pour chacun des niveaux du facteur `z`.
`aggregate(y ~ z, data=, cmd, ...)`	Application d'une commande `cmd` à une variable numérique `x` pour chacun des niveaux du facteur `z`.
`t.test(x, y, ...)`	Test de Student en supposant l'égalité des variances (option `var.equal = TRUE`) ou non, pour échantillons indépendants ou appariés (option `paired = TRUE`).
`wilcox.test(x, y, ...)`	Test de Wilcoxon pour échantillons indépendants ou appariés (option `paired = TRUE`).
`lm(y ~ z, data=, subset=, ...)`	Modèle linéaire (analyse de variance à un ou plusieurs facteurs à effets fixes). Cette commande permet d'estimer les paramètres du modèle.
`drop1(mod, test=, ...)`	Comparaison de modèles emboîtés, permettant de construire les tests de Fisher-Sndecor (option `test = "F"`) pour les effets principaux.
`aov(y ~ z, data=, subset=, ...)`	ANOVA à un ou plusieurs facteurs à effets fixes, avec ou sans mesures répétées (option `Error=`).
`model.tables(mod, ...)`	Résumé de synthèse des effets (écarts des moyennes de groupe à la moyenne générale) sous forme de tableau.
`plot.design(y ~ z, data=, fun=)`	Résumé de synthèse des effets (écarts des moyennes de groupe à la moyenne générale) sous forme de graphique.
`replications(y ~ z, data=)`	Nombre d'unités statistiques allouées dans chaque traitement (croisement des niveaux de deux ou plusieurs facteurs), incluant les interactions.
`pairwise.t.test(x, z, ...)`	Tests de Student pour des paires de moyenne (cas indépendants ou appariés), avec correction des degrés de significativité (option `p.adjust.method=`).
`pairwise.wilcox.test(x, z, ...)`	Tests de Wilcoxon simultanés pour des paires de moyenne (cas indépendants ou appariés), avec correction des degrés de significativité.

7.2 Tableaux de contingence

Commande	Description
`table(z1, z2, useNA=, ...)`	Tableau d'effectifs pour une ou deux variables. L'affichage des valeurs manquantes est géré par l'option `useNA = "always"`.
`prop.table(tab, margin=)`	Fréquences relatives d'un tableau créé par `table()` par rapport à l'effectif total, ou aux totaux lignes (`margin = 1`) ou colonnes (`margin = 2`).
`margin.table(tab, margin=)`	Calcul des totaux lignes (`margin = 1`) ou colonnes (`margin = 2`) d'un tableau de contingence construit à l'aide de `table()`.
`chisq.test(tab, ...)`	Test du chi-deux pour un tableau de contingence construit à l'aide de `table()`.
`fisher.test(tab, ...)`	Test exact de Fisher pour un tableau de contingence construit à l'aide de `table()`. Usage alternatif : `fisher.test(z1, z2, ...)`.
`mcnemar.test(z1, z2, ...)`	Test de McNemar pour un tableau de contingence construit à l'aide de `table()`, dans le cas de deux échantillons appariés.
`pairwise.prop.test(tab, ...)`	Tests multiples de deux proportions, avec correction des degrés de significativité (option `p.adjust.method=`, par défaut méthode de Bonferroni).

7.3 Corrélation et régression linéaire

Commande	Description
`cor(x, y, method=, use=, ...)`	Calcul du coefficient de corrélation de Pearson (ou Spearman, avec `method = "spearman"`). L'option `use = "pairwise"` est utile en cas de données manquantes.
`cor.test(x, y, method=, use=, ...)`	Test de significativité du coefficient de corrélation linéaire de Pearson (ou Spearman, avec `method = "spearman"`).
`lm(y ~ x, data=, subset=, ...)`	Modèle linéaire (régression linéaire simple et multiple). Cette commande permet d'estimer les paramètres du modèle.
`summary(mod)`	Tableau des coefficients de régression et test de Student associés pour un modèle de régression linéaire à un ou plusieurs facteurs.
`anova(mod, ...)`	Tableau d'analyse de variance associé à un modèle de régression linéaire à un ou plusieurs facteurs.
`coef(mod)`	Valeurs des coefficients de régression pour un modèle de régression linéaire à un ou plusieurs facteurs.
`fitted(mod, ...)`	Valeurs prédites pour l'ensemble des observations utilisées lors de la construction du modèle de régression.
`resid(mod, ...)`	Valeurs résiduelles (différence entre valeurs observées et valeurs prédites) d'un modèle de régression linéaire.
`predict(mod, newdata=, ...)`	Valeurs prédites pour un modèle de régression à partir de données non nécessairement observées. Par défaut, cette commande renvoie les valeurs ajustées.

7.4 Régression logistique

Commande	Description
`glm(y ~ x, data=, subset=, family=, ...)`	Modèle linéaire généralisé (p. ex., régression logistique, `family = binomial(logit)`). Cette commande permet d'estimer les paramètres du modèle.
`summary(mod)`, `anova(mod, test=, ...)`	Identique au cas de la régression linéaire simple ou multiple (tableau d'analyse de déviance au lieu de variance).
`coef(mod)`, `predict(mod, newdata=, type=, ...)`	Coefficients de régression et valeurs prédites sur l'échelle du log-odds ou sous forme de probabilités individuelles (`type = "response"`).

7.5 Données de survie

Commande	Description
`Surv(time=, event=, ...)`	Construction d'une variable censurée associant un temps d'observation (`time=`) à un statut relatif à l'événement d'intérêt.
`survfit(s ~ 1, ...)`	Médiane de survie et intervalle de confiance à 95~% dans le cas d'un (`s ~ 1`) ou plusieurs (`s ~ z`) échantillons.
`summary(s, times=, ...)`	Tableau des valeurs de la fonction de survie (l'option `times=` permet de limiter l'affichage à certaines valeurs de temps).
`survdiff(s ~ z, data=, ...)`	Test d'égalité de 2 ou plusieurs fonctions de survie par le test du log-rank (par défaut) ou de Gehan-Wilcoxon (`rho = 1`).
`plot(s, conf.int=, fun=, ...)`	Courbe de Kaplan-Meier pour un ou plusieurs échantillons, avec IC à 95 % (utiliser `conf.int = 0` pour les supprimer).
`coxph(s ~ z, data=, subset=, strata=, ...)`	Modèle de régression de Cox (risques proportionnels), éventuellement stratifié sur une variable de type facteur (`strata=`).
`summary(mod)`, `anova(mod)`, `coef(mod)`	Tableau des coefficients de régression avec tests de Wald associés et tableau d'analyse de déviance (statistique du chi-deux).